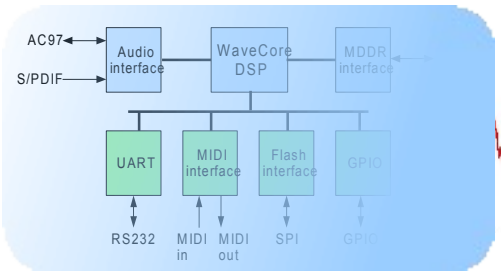
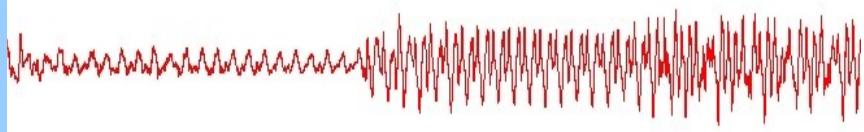
Product brief

WaveKick audio/acoustics development platform





Feature summary

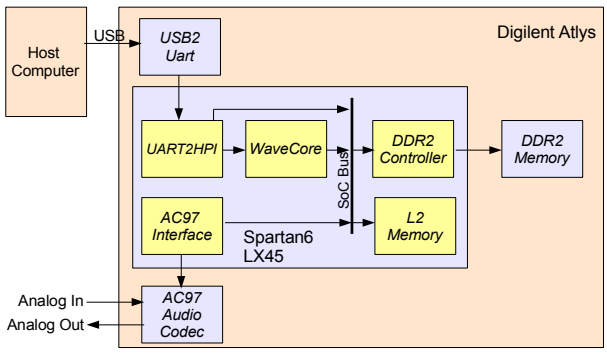
- Single-precision floating-point DSP technology
- Very efficient compiler: only 5 instructions for a BiQuad filter
- Integrated support for variable-length delay-lines
- Powerful graphical and audible debug features
- Graphical hierarchical programming methodology
- Versatile DSP SW library available: basic building blocks and complete audio processing blocks
- Complete core abstraction for the programmer
- Multi-rate and multi-channel stream processing supported
- Embedded support for IIR and FIR filter generation
- Run-time manipulation of parameters supported
- Fast compiler: few seconds to compile a modestly complex audio DSP algorithm
- Fast simulator: real-time execution of algorithms such as 12-stage phaser
- Virtual zero latency processing
- Huge performance: 860 MFLOPS sustained

Product abstract

WaveKick is a DSP development kit which is specifically targeted to audio/acoustical processing and sound synthesis. The platform is easy to program at a convenient abstraction level and the programmer does not have to posses classical programming skills (like C or C++).

The kit consists of:

- A processor hardware platform. This is a commercial off-the-shelf Digilent Atlys FPGA platform which is powered by *WaveCore* processor technology.
- A powerful and very efficient IDE, which consists of a compiler, a simulator and associated debug tools.
- A comprehensive and versatile *WaveLib* DSP SW library. This library contains all kinds of basic signal processing building blocks, like oscillators, function generators, linear filters, etc. Furthermore, the library contains dedicated audio processing function blocks, like phaser, flanger, graphic EQ, frequency pitch shifter, etc.



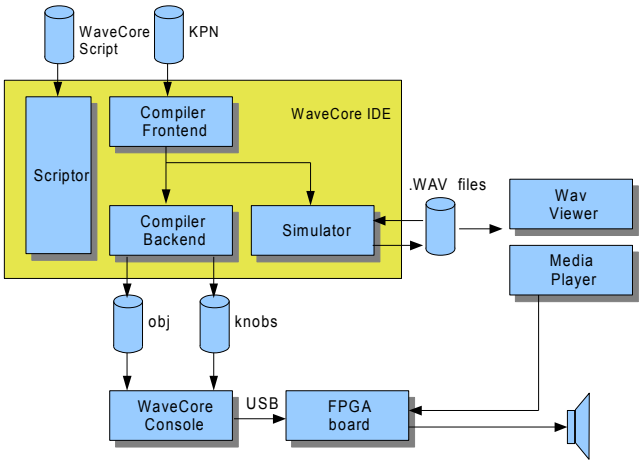
The *WaveCore* processor is equipped with 128MByte of DDR2 memory which is used as delay-line memory. The processor communicates with a host computer system through USB (loading object code to the *WaveCore* processor and/or run-time manipulation of a *WaveCore* application. The processor board has analog I/O and can either be connected to a host computer system or to guitar and guitar-amp or other audio devices.

Product brief

WaveKick audio/acoustics development platform

WaveCore IDE

The *WaveCore* programming methodology is based on graphical design entry, and is similar to Simulink (MathWorks). The programmer describes one or more (interconnected) Signal Flow Graphs (SFG). Furthermore, the programmer is enabled to route an arbitrary number of primary input signals to arbitrary SFG entry points. Similarly, the programmer can probe on arbitrary nodes within the SFG. The programmer can also define signals which can be linked to “knobs”, which implies that these are made run-time controllable by means of the HW console.



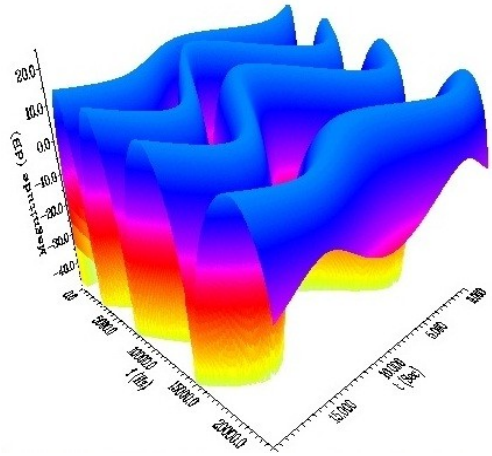
The *WaveCore* IDE basically consists of three main components:

- A simulator engine, plus associated post-processor. The programmer is enabled to probe into the SFG on arbitrary nets, using an arbitrary number of probes. Each probe can act as signal source or sink, and is linked to a multi-channel WAV file. The post-processor links probed signals to a configurable number of display windows. A WAV file can also be played by a media player.
- A compiler, which consists of a front-end and a back-end. The front-end translates the SFG description to a *WaveCore* DSP independent C-type datastructure. The back-end translates the constructs to *WaveCore* DSP instructions.
- A *WaveCore* HW console, which loads the generated object code to the processor (or Flash memory). Furthermore, the HW console can act as a run-time controller for the application which runs on the *WaveCore* DSP. The compiled “knobs” can be run-time manipulated by the user through the HW console.

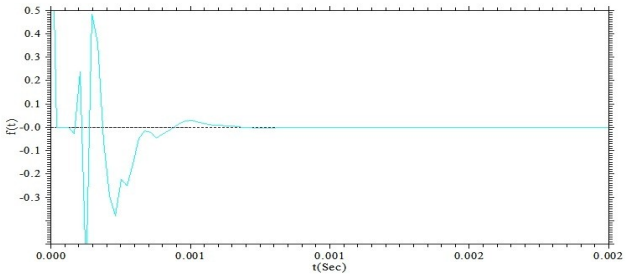
The three mentioned main components within the IDE can either be controlled by the GUI, or by a script.

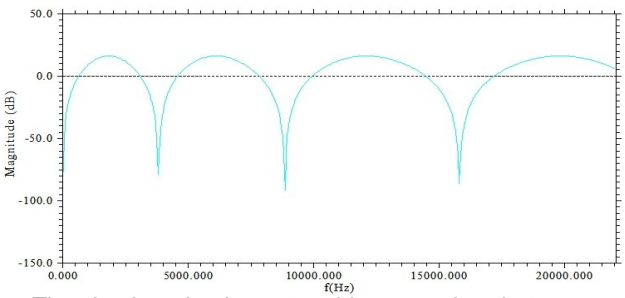
Probed signals are either routed to (multi-channel) WAV file during simulation, or to a physical DSP port when compiled to object code. Similarly, a (multi-channel) WAV file can be linked to a source-probe and injected into the SFG during simulation. In the same way one or multiple signals can be routed from a physical DSP input port to the SFG when compiled to object code.

The simulator is thus enabled to feed an arbitrary number of input signals from WAV-files to the SFG, and similarly to route an arbitrary number of signals to WAV-files.



The post-processor reads the probed signals from WAV file(s) and displays them in either time-domain, frequency domain (FFT) or time/frequency domain (spectrogram).





The simulator implements a bit-true and cycle-true representation of the *WaveCore* DSP. Hence it is not necessary to support on-chip debug of a compiled DSP program.

Product brief

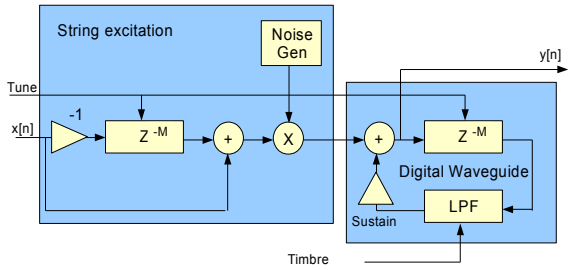
WaveKick audio/acoustics development platform

Performance benchmark

The *WaveKick* platform has been benchmarked against an Intel Core i3 (on a Windows computer). This has been done by comparing the *WaveCore* simulator, which runs on the Intel processor, with the *WaveCore* processor on the Atlys board. The benchmark analysis consists of 5 subsequent steps, called Lab1 – Lab5. The sampling frequency is 48kHz for all these labs.

Lab1: guitar string model

The benchmark algorithm is based on a Digital WaveGuide (DWG) model which represents a guitar string. An SFG-style algorithm description of this DWG algorithm is depicted in the following figure.



This string model consists of a “string excitation” part, which generates a noise burst with length M on a signal edge at the input x[n]. This noise burst is injected in the DWG model where a variable conrolled lowpass filter (LPF) models damping of the plucked string.
The associated *WaveCore* program consists of only 9 lines of code (see *WaveCore* programmers manual for details on the language):

```
SFG
.Input stage: pulse synthesis
M .x[n] void .x[n-1000]  -1 1000
. Subtract delayed copy of x from x[n-1000]
. This yields a pulse at rising and
. and falling edge:
A .x[n] .x[n-1000] p[n] 0 1
.
. Create white noise burst:
R void void Noise 0.2 0
* Noise p[n] NoiseBurst 0 1
.
. Karplus-Strong:
A NoiseBurst Lpf_.y[n] .y[n] 0 1
M .y[n] void Lpf.x[n] 1 1000
INSTANCE ..\SFGs\BasicCircuits\vclpf.sfg Lpf
M .Timbre void Lpf.f[n] 1 1
* Lpf.y[n] .Sustain Lpf_.y[n] 0 1
.
. Tuning: modulation of delay-line length:
B .Tune Lpf.x[n] DummyOut 0 1
B .Tune .x[n-1000] DummyOut1 0 1
GFS
```

A single-string model is compiled to the *WaveKick* kit and also simulated with the *WaveCore* simulator. The results are displayed in the performance benchmark table.

Lab2: 6-string model

In the Lab2 algorithm, 6 string instantiations have been put together in a “6-string” model, and tuned according to an e-chord. This model is also compiled to the *WaveKick* kit and compared with simulation.

Lab3: guitar model

In the Lab3 algorithm, the 6-string model of Lab2 is extended with an acoustical model of a guitar body, based on a 1500-taps FIR filter. This makes a realistic model of an acoustical guitar with run-time control of tuning parameters of each individual string, plus damping characteristics.

Lab4: guitar model plus 12-stage phaser

The guitar model of Lab3 is extended with a 12-stage phaser which run-time control variables are added to the ones from lab3.

Lab5: large FIR

This algorithm consists of the 6-string model, a 6000-taps FIR filter, and the 12-stage phaser. This algorithm occupies 70% of the *WaveKick* processing capacity.

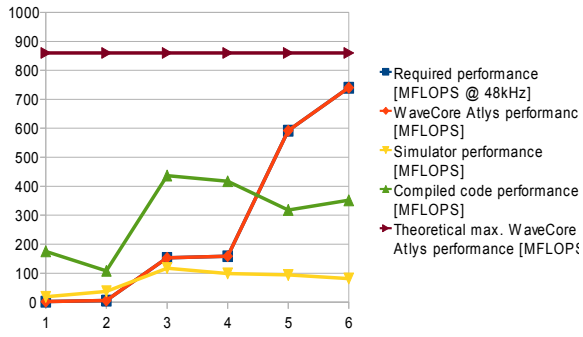
Lab6: Interconnected Biquad filters

This benchmarks consists of 1710 interconnected biquad filters. This heavy-duty algorithm with complex interconnect structure occupies 86% of the *WaveKick* processor capacity.

The following table compares *WaveKick* with the *WaveCore* simulator performance and optimized C-code (which optionally can be generated by the *WaveCore* compiler). The simulator and optimized C-code run on an Intel core I3 processor.

	Algorithm complexity [#FP operations/iteration]	Required performance [MFLOPS @ 48kHz]	WaveCore Atlys performance [MFLOPS]	Simulator performance [MFLOPS]	Compiled code performance [MFLOPS]
Lab1	38	1,82	1,82	18,24	175,39
Lab2	117	5,62	5,62	37,44	108
Lab3	3186	152,93	152,93	117,56	436,8
Lab4	3303	158,54	158,54	99,09	417,17
Lab5	12330	591,84	591,84	93,71	318,11
Lab6	15420	740,16	740,16	81,73	351,58

Note: the *WaveCore* simulator or compiled C-code only occupies 50% of one of the dual-core Intel core processor. From this table the following performance comparison graph is derived:



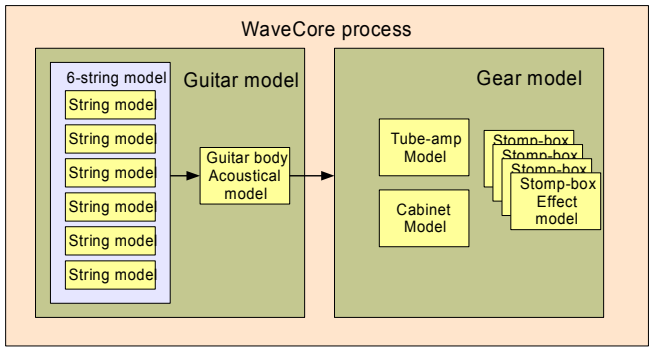
The numbers 1 to 6 on the x-axis refer to Lab1-Lab6. A fully-loaded *WaveCore* processor on the *WaveKick* kit can execute up to 860 MFLOPS. This implies that the *WaveKick* kit outperforms the Intel core i3, even with compiled optimized C-code of the Lab algorithms.

Product brief

WaveKick audio/acoustics development platform

WaveLib DSP SW library

A versatile and comprehensive DSP SW library is available within the *WaveKick* development kit.
Each DSP SW building block can be instantiated and combined with any other unit in any arbitrary way, and synchronized to an arbitrary multiple of the applied sampling frequency. The library consists of basic units, and specific audio processing units.



Examples of basic DSP SW units are:
- Oscillators (single-phase, quadrature, sinusoidal, triangular, square, sawtooth)
- Dynamic range compressor
- Dynamic range limiter
- Noise-gate
- IIR filters: lowpass, highpass, bandpass, bandrejection, allpass, Butterworth, Chebychev, Bessel, order [1..10]
- FIR filters: impulse-response reader automatically generates convolver units
- Digital WaveGuide guitar string model
- Guitar model
- Low Frequency Oscillator (LFO)

Furthermore, example *WaveCore* guitar effect programs are available:
- 12-stage phaser
- Flanger
- Chorus
- Echo
- Reverb
- WahWah
- Distortion
- Tremolo
- Cabinet simulator

About Mathlon Technology

Mathlon Technology is a young company, which has been founded in 2011.
The company benefits from over 20 years of experience within several electronics engineering fields, such as DSP technology development specifically for the domain of audio and GSM.

Apart from the products which are described in this product brief, Matlon Technology also delivers consultancy and design services within the domain of DSP

Contact information:

Mathlon Technology
Gastendonkstraat 20
5961 JX Horst
The Netherlands
+31 (0)6 46 33 15 76
info@mathlon.nl
www.mathlon.nl
Kvk: 53524276
BTW: NL159992175B01
ING rek. Nr: 6925879